METHOD AND SYSTEM OF CONTROLLING A CONTEXT MENU

CROSS-REFERENCE TO RELATED APPLICATION

[001] This application claims the benefit of U.S. Provisional Application Serial No. 60/493,029, filed August 5, 2003, entitled METHOD AND SYSTEM OF CONTROLLING A CONTEXT MENU, which is hereby incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[002] The present invention is directed to computer interfaces and, more particularly, to a method and system of controlling a context menu.

2. Description of Related Art

[003] When accessing the Internet (i.e., the worldwide Web, the Web, etc.), an Internet user typically executes, via a computer, a browser software program such as, for example, Netscape Navigator™ or Microsoft Internet Explorer™. The browser program (i.e., a browser) establishes a link to the Internet (via a modem and an Internet Service Provider (ISP), for example) and also provides a textual and graphical user interface (i.e., an application window) and a window within the application window for displaying Internet content (i.e., a browser window).

[004] While using the browser program, a user is able to call up a context menu, e.g., by clicking on the right mouse button or activating a context menu key. The context menu is comprised of a list of items that can be chosen, typically, to perform different browser functions. Different context menus are displayed depending on what was clicked on and/or selected by the user in the browser program. The context menu gives a user quick access to browser functions that are related to the element that was clicked, making Internet navigation more convenient. For these reasons, it is often desirable to insert additional menu items to a context menu that add functionality to the browser program.

[005] Current browser programs allow an application to add text, and to associate handlers with the text, in context menus, by modifying registry settings. This method of modifying context menus, however, has several shortcomings. One such shortcoming is that, with this method, the application has no control over the order and position of the additional items relative to standard or other menu items in the context menu. Another shortcoming of current context menu modification techniques is that they offer the same menu items regardless of the user using the browser, and regardless of the country, etc., in which the user is located during use of the browser. Still another shortcoming is that updating or changing context menu items requires the user to install a new application at the user computer. As such, a need exists for an improved system and method of controlling a context menu.

SUMMARY OF THE INVENTION

[006] The invention satisfies these and other needs, as will be apparent from the teachings herein. An embodiment of the invention, for controlling a context menu, can comprise downloading additional menu items that are not part of the context menu prior to download, and monitoring for when a user calls for a context menu. Upon detecting a context menu call, it is determined what region was selected by the user, and at least one of the additional menu items is made available to the user as part of the context menu interface.

[007] Another embodiment of the invention is directed to a method of controlling a context menu provided by a company, used with a Web page. The method comprises downloading additional menu items that are not part of the context menu prior to download, and monitoring, via a client-side module, for when a user calls for a context menu. Upon detecting a context menu call, at least one of the additional menu items is made available to the user as part of the context menu interface, and the placement of the additional menu items is determined by the client-side module.

[008] Another embodiment of the invention is directed to a method of controlling a context menu, used with a Web page, and comprises downloading additional menu items and menu icons, via a client-side module, that are not part of the context menu prior to download, and monitoring for when a user calls for a context menu. Upon detecting a context menu call, at least one of the additional menu items is made available to the user as part of the context menu interface, and at least one of the icons are placed adjacent to at least one of the additional menu items.

[009] Other objects and features of the present invention will become apparent from the following detailed description, considered in conjunction with the accompanying drawing figures. It is understood, however, that the drawings are designed solely for the purpose of illustration and not as a definition of the limits of the invention, for which reference should be made to the appended claims.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

[0010] In the drawing figures, which are not to scale, and which are merely illustrative, and wherein like reference numerals depict like elements throughout the several views:

[0011] FIG. 1 is a schematic illustrating a system implemented according to an embodiment of the present invention;

[0012] FIG. 2 is a flowchart illustrating the process of controlling a context menu according to an embodiment of the present invention;

[0013] FIG. 3 is a flowchart illustrating the process of determining what element a user selected according to an embodiment of the present invention;

[0014] FIG. 4 is a schematic illustrating an exemplary HTML structure;

[0015] FIG. 5 is a flowchart illustrating the process of providing computer code for controlling a context menu according to an embodiment of the present invention;

[0016] FIG. 6 is an exemplary screen shot illustrating an exemplary context menu, according to an embodiment of the present invention;

[0017] FIG. 7 is an exemplary screen shot illustrating another exemplary context menu, according to an embodiment of the present invention;

[0018] FIG. 8 is an exemplary screen shot illustrating another exemplary context menu, according to an embodiment of the present invention; and

[0019] FIG. 9 is an exemplary screen shot illustrating another exemplary context menu, according to an embodiment of the present invention.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

[0020] There will now be shown and described in connection with the attached drawing figures several exemplary embodiments of a system and method of controlling a context menu.

[0021] With reference to FIG. 1, there is shown a block diagram of a system 100 implemented in accordance with certain embodiments of the invention. A computer 150 is connected to the Internet 190. Computer 150 comprises an internal bus 164 that facilitates communication of information (i.e., digital data) between and among the various devices of the computer 150 and that also facilitates communication between the computer and external devices and systems via a communication interface 168. A processor 166 coupled to the bus 164 processes information within the computer 150. The computer 150 also includes a memory 160 such as, for example, Random Access Memory (RAM) and/or other equivalent dynamic memory storage devices, coupled to bus 164 for receiving and storing instructions communicated from the processor 166. Memory 160 may also be used to temporarily store variable or other intermediate information while the processor 166 executes instructions. Read-Only-Memory (ROM) 162 is also coupled to the bus 164 for storing static data and instructions for use by the processor 166.

[0022] Various input and output devices are provided as part of computer 150, including, by way of non-limiting example, a display 154 (e.g., cathode ray tube (CRT), liquid crystal display (LCD), etc.), an input device 156 such as a keyboard, and a cursor control device 158 such as a mouse, or trackball, for example. A data storage device 152 such as, for example, a magnetic disk drive and magnetic disk, a CD-ROM drive and CD-ROM, or other equivalent devices and data storage mediums, is coupled to the bus 164 for communication with the processor 166, main memory 160, and communication interface 168. The storage device 152 preferably has an operating system 170 and an Internet browser software program 172 (i.e., a browser) stored thereon. As will be discussed in greater detail below, a client-side module 174 may also be stored on the data storage device 152.

[0023] The computer 150 may communicatively connect to the Internet 190 via the communication interface 168 over one or more transmission media including, but not limited to, coaxial cable, copper wires, and fiber optical cables. Communication between the computer 150 and the Internet 190 may also be via a wireless or cellular interface. The communication interface 168 facilitates two-way communication between the computer 150 and another electronic device or system, e.g., a server computer (not shown) provided by a content provider 120, 130.

[0024] An Internet user (not shown) using the computer 150 may gain access to the Internet 190 by causing the browser 172 to execute, thereby opening a communication link between the communication interface 168 of the computer 150 and an Internet site 126 of content provider 120, via an Internet Service Provider (ISP) 180. Internet content is communicated by the content provider 120 to the computer 150 for display by browser 172. Alternatively, a content provider 120, 130 may also be and ISP 180.

[0025] In alternative embodiments, computer 150 may be a desktop or notebook computer, PDA, hand held device, or wireless phone (with graphics capability), or any other device now known or hereafter developed that is capable of performing the functions as described herein.

[0026] In accordance with an embodiment of the invention, a first Internet content provider 120 may provide an Internet user with access to a program 122 for controlling the browser 172. When executed by the user, the controlling program 122 downloads or creates a client-side module 174 such as, for example, a Dynamic Link Library (DLL), on the data storage device 152 of the Internet user's computer 150. The client-side module

174 preferably includes ActiveX control or Plug-in functionality. Thereafter, when the Internet user accesses the Internet using the browser 172, the browser 172 opens the client-side module 174 and preferably automatically (or otherwise) establishes a connection to the content provider's Internet site 126. The content provider, in response to the connection established by the browser 172, loads information and/or functional data into a shell operating within the browser and created by the client-side module 174. For example, if the user has an account with the content provider 120, customized information and/or functionality may be loaded into the client-side module 174. If the user does not have an account, more generalized (e.g., guest) information and/or functionality may be loaded.

[0027] In certain embodiments, the client-side module 174 essentially opens a shell (or a plurality of shells) within the browser 172 that contains the ActiveX control or Plug-in code that may control, i.e., add, remove, and/or modify, the Internet browser 172. When loaded with the ActiveX control or Plug-in, the client-side module 174 preferably contains functions, objects, data, and other software, referred to generally herein as information, that may be used to control the browser 172. The present embodiment ensures that the client-side module 174 (and shell) does not close when the Internet user moves, for example, from Internet site 126 (having a Web page 124) to Internet site 136 (having Web page 134). Thus, the information and/or functionality provided via the ActiveX control or Plug-in is not lost when the Internet user disconnects from the Internet site that loaded the ActiveX control or Plug-in, and connects to another Internet site. In alternate embodiments the client-side module 174 may operate “behind” the browser, and appear invisible to the user until a context menu is called. Still in other embodiments, client-side module 174 may be located at a remote location from the Internet user.

[0028] FIG. 2 illustrates an exemplary embodiment of a method 200 for controlling a context menu. The method 200 starts with step 205 with the client-side module 174 transmitting a request to a predetermined Internet site, e.g., content provider 120, where the request is received, for a description of the context menu items. This request can be part of and/or implied in the connection established between the client-side module 174 and the content provider's Internet site 126 as mentioned above.

[0029] Next, in step 210, the client-side module 174 receives data from the predetermined Internet site 126, the data including descriptions of context menu items. In one exemplary example, wherein the method of controlling a context menu is implemented with a client-side module 174 for a browser toolbar, the context menu item descriptions are received by the client in the same feed as the regular buttons of the toolbar. The description can include, for example, types, URLs, tiles, mnemonic keys, and start positions.

[0030] Processing proceeds from step 210 to step 215, where the client-side module 174 determines when the browser 172 has finished loading a Web page. This may be done, in one exemplary embodiment, by coordinating events on the Web browser control so that the Web browser notifies the client-side module 174 when different events happen. For example, when a Web page is complete, the client-side module 174 will receive a DISPID_DOCUMENTCOMPLETE event.

[0031] Returning to the description of method 200, in step 220, the client-side module 174 monitors for context menu events. In one exemplary embodiment, after a

Web page is finished loading, the client-side module 174 retrieves the IHTMLDocument2 interface for the page. Using an IHTMLDocument2 pointer, the client-side module 174 can monitor events that are related to the HTML document. In particular, the client-side module 174 looks for the event called DISPID_HTMLDOCUMENTEVENTS2_ONCONTEXTMENU, which is initiated, or in the computer programming vernacular, "fired" when the user clicks or activates the right mouse button (or other appropriate selector) when the cursor is positioned in the browser window.

[0032] Next, the client-side module 174 determines if a context menu event occurred, in step 225. If no context menu event occurs, processing returns to step 220, where the client-side module 174 continues to monitor for context menu events. Returning to decision step 225, if the client-side module 174 determines that a context menu event has occurred, e.g., a DISPID_HTMLDOCUMENTEVENTS2_ONCONTEXTMENU event occurs, processing proceeds to step 230.

[0033] In step 230, the client-side module 174 prepares the browser window to view and possibly modify the events received by the browser window related to the context menu. In a certain embodiment, when the client-side module 174 detects a DISPID_HTMLDOCUMENTEVENTS2_ONCONTEXTMENU event, the client-side module 174, subclasses the browser window. Subclassing the browser window allows the client-side module 174 to view the events received by the browser window. Subclassing may be performed, for example, by using a CContainedWindow object that directs messages to the client-side module 174. To determine which window the client-side module 174 will subclass, the client-side module 174 walks down the window hierarchy starting at the top level browser window looking for a window with the classname of "Internet Explorer_Server." Once identified, that window will be subclassed. Browser window subclassing of this type is described in further detail in U.S. Patent Application Serial No. 09/429,585, filed October 28, 1999, and entitled "A Method Of Controlling An Internet Browser Interface And A Controllable Browser Interface," the contents of which are hereby incorporated by reference herein.

[0034] Returning to the description of method 200, in step 235, the client-side module 174 determines what element on the Web page the client selected, and what text (if any) was highlighted when the client called for, or activated, the context menu. Step 235 is described in greater detail below with respect to method 300 and with reference to FIG. 3. Referring back to FIG. 2, after step 235, the client-side module 174 will have obtained information related to the selected element, including the type of the element, which may be a combination of at least an anchor (e.g., a clickable hyperlink), an object (e.g., an ActiveX control embedded on the Web page), an image (e.g., a picture embedded on the page), and text; the URL that was selected, if any; and what text was selected (or highlighted) at the time, if any. With the element information, selected text and having prepared the browser window in step 230, processing proceeds to step 240.

[0035] In step 240, the client-side module 174 monitors events before they are processed by the browser window. In one exemplary embodiment, the client-side module 174 looks for messages including the messages WM_INITMENU, WM_EXITMENULOOP, WM_MENUCHAR, WM_MENUSELECT, WM_MEASUREITEM, and WM_DRAWITEM.

[0036] When a message indicating the context menu is about to be displayed is received by the client-side module 174, processing proceeds from monitoring step 240 to step 245, where the client-side module 174 adds menu items to the context menu based upon what element, if any, the client selected and what text, if any, was selected. In one exemplary embodiment, a WM_INITMENU message indicates that a context menu is about to be displayed. When the client-side module 174 receives a WM_INITMENU message, the client-side module 174 identifies the type of the element that was selected so that the client-side module 174 can determine where in the context menu to place the additional menu items. In certain embodiments, the starting position for each type of element is predetermined and is included in the context menu item description received from the predetermined Internet site. If an element has more than one type, the order of precedence is Anchor, Image, Text (e.g., if type is Anchor and Image, the client-side module 174 starts at the Anchor starting position). In alternate embodiments, different orders of precedence, or no particular order of precedence, may be employed. Once the client-side module 174 determines the starting position, the client-side module 174 adds menu items to the context menu using, in certain embodiments, Windows Application Programming Interfaces (APIs). If there were one or more mnemonic characters in the received context menu descriptions, the client-side module 174 stores the mnemonic characters when the client-side module 174 adds the menu item. Furthermore, the additional or original context menu items can also include submenus. For example, when a user floats a pointer over and/or selects the additional or original context menu item, a second context menu, giving a user additional menu items, can be created. In addition, alternatively, an item on the second context menu can also lead to a third context menu, and so on. Such submenus could be employed to help organize related items.

[0037] When a message indicating the context menu is dismissed is received, e.g., the client chooses a menu item or clicks somewhere outside the context menu, processing proceeds from step 245 to step 250, where the client-side module 174 removes the added menu items from the context menu. In a certain embodiment, a WM_EXITMENULOOP message indicates that a context menu is dismissed. When the client-side module 174 receives a WM_EXITMENULOOP message, the client-side module 174 removes the added context menu items.

[0038] In one exemplary embodiment if the client-side module 174 receives a WM_MENUCHAR message, this would indicate that the client had hit, or depressed, a keyboard key while the context menu was displayed. In this case, the client-side module 174 compares the key that was hit to the mnemonic keys (with the comparison being case insensitive) of the context menu items that were added. If there is a match, the context menu is dismissed, and a WM_MENUSELECT message is sent to the client-side module 174 with values that indicate to the client-side module 174 that the context menu was dismissed.

[0039] Returning to the description of method 200, processing proceeds from step 250 to decision step 255. In step 255, the client-side module 174 determines if an item was selected. In one exemplary embodiment, a WM_MENUSELECT message indicates that a user either hit a mnemonic key corresponding to a menu item or made a selection with the mouse. A WM_MENUSELECT message can also indicate when a user drags the mouse to point at a menu item, e.g., so that the menu item can be highlighted.

[0040] If the client-side module 174 determines that a selection was made, then processing proceeds from step 255 to step 260, where the selection is processed. When a WM_MENUSELECT message is received with a parameter indicating that the context menu has been dismissed, and the user chose a context menu item either by mouse or by keystroke (or otherwise), then a selection was made and it is processed. In one embodiment, the client-side module 174 looks at the context menu items description for the data that corresponds to the item selected. From this data, the client-side module 174 substitutes the default URL with data that is specific to the current Web page, such as the URL or text selection. Once the information is substituted, it is processed.

[0041] Returning to decision step 255, if it is determined that a context menu item was not selected, processing proceeds to step 265. In step 265, the client-side module 174 determines if the browser is still active. If the browser is active, processing returns to step 220 and the client-side module 174 monitors for context menu events. If, on the other hand, the client-side module 174 determines that the browser is no longer active, processing of method 200 is terminated.

[0042] Turning to FIG. 3, with continued reference to FIG. 1, there is shown an exemplary embodiment of the method 300, for determining what element the user selected and what text was selected when the user called for the context menu. Processing of method 300 begins at step 310, where the client-side module 174 identifies the element on the Web page that was selected by the user. This element is considered to be the source element. In an embodiment, the client-side module 174 can identify the source element by using the IHTMLEventObj pointer that corresponds to the selection (e.g., click), in question. This pointer contains information about the click event, including what HTML element was selected (clicked on). Processing then proceeds from step 310 to step 315.

[0043] In step 315, the client-side module 174 determines if the current element is valid. In certain embodiments, the client-side module 174 determines if the current element is an Anchor, Object, or Image by using ATL smart pointer CComQIPtr<Class>. For example, to test the element as an ANCHOR, the client-side module 174 can use the following call:

```
CComQIPtr<IHTMLAnchorElement> pAnchor = pElem;
```

If this call succeeds and pAnchor is not NULL, i.e., has a value, the client-side module 174 determines that the element is an ANCHOR. Similar processes are used to determine if an element is an Object or an Image. Returning to method 300, if, in step 315, the client-side module 174 determines the current element is valid, processing proceeds to step 320, where the client-side module 174 determines if the type of the element is a type of interest, for example, in one embodiment of the present invention, anchor and image type are of interest. In other embodiments, other types such as Object and Text may also be of interest and addition context menu items may be added giving a user additional functionality.

[0044] In step 320, if the client-side module 174 determines the type of the current element is not of interest, processing proceeds to step 335, where the client-side module 174 walks up the Web page (e.g., HTML document), structure and identifies the next parent element, i.e., the preceding element. Processing proceeds from step 335 to decision step 315, where the client-side module 174 determines if the current element is valid. If the current element is valid, processing proceeds from step 315 to step 320. In step 320, if the client-side module 174 determines the type of the current element is a type of interest, proceeding proceeds from step 320 to step 325.

[0045] In step 325 the client-side module 174 determines if the current element is an image. If the client-side module 174 determines that the current element is not an image processing proceeds to step 330, where the client-side module 174 saves the element, including related information (e.g., the type of the element and the associated URL). From step 330, processing returns to step 335, where the next parent is identified. Steps 315 and 320 are repeated as described above. In step 325, if the client-side module 174 determines the current element is an image, processing proceeds to step 345.

[0046] In step 345, the client-side module 174 determines if the image is an image map. As is known to those skilled in the art, an image map is an image that is subdivided into clickable areas that can each map to different URLs when clicked. The URL can be a bookmark into the current page, or a link to another Web page. In certain embodiments, the client-side module 174 looks at the "useMap" property of the image element. If the value of the property is blank, e.g., "", the image is not an image map. If the ”useMap” property of the image is a URL, then the image is an image map. Returning to step 345, if the client-side module 174 determines the image is not an image map, processing proceeds to step 350, where the current element is saved. From step 350, processing proceeds to step 355, where the next parent element is identified. Steps 315, 320, and 325 are repeated as described above.

[0047] Returning to step 345, if the client-side module 174 determines the image is an image map, processing proceeds to step 355, where the client-side module 174 obtains the data related to the image map. In one exemplary embodiment, the image map data is stored as a set of coordinates and URLs, e.g.:

```
<MAP NAME="map1">
<AREA NAME="area1" COORDS="0,0,99,100"
    HREF="http://www.yahoo.com/" TARGET="frame1">
<AREA NAME="area2" COORDS="100, 0,199, 100"
    HREF="http://sports.yahoo.com/" TARGET="frame1">
<AREA NAME="area3" COORDS="200,0,299,100"
    HREF="http://weather.yahoo.com/" TARGET="frame1">
<AREA NAME="area4" COORDS="300,0,400,100"
    HREF="http://companion.yahoo.com/" TARGET="frame1"> </MAP>
```

[0048] The COORDS describe the bounding rectangle of that map area and the HREF is the URL that the browser will navigate to when the user clicks in the described area. This image map information is stored for later use.

[0049] Returning to the description of method 300, processing proceeds from step 355 to step 360. In step 360, the client-side module 174 determines what part of the map the user selected, (clicked on). In one embodiment, the client-side module 174 determines what part of the map a user clicked on by first calling get_all() on the IHTMLDocument2 to get a collection of all HTML elements on the page. Then, the client-side module 174 finds the element with a name that is the name of the map the image is using. Then the client-side module 174 determines the offset x,y of the user's click relative to the upper-left corner of the source element, i.e., the image element. Then the client-side module 174 gets a collection of all AREA elements contained in the MAP. For each AREA element in the collection, the client-side module 174 determines if the

X,Y (or coordinates) of the user's click lies within the bounding rectangle described by the AREA's COORDS. Once the client-side module 174 determines which AREA the user clicked on, the client-side module 174 can get the the HREF property.

[0050] Returning to the description of method 300, processing proceeds from step 360 to step 330, where the element, including related information such as type and URL, is saved for later use. Processing proceeds from step 330 to step 335, where the next parent element is identified. Then, processing proceeds from step 335 to step 315.

[0051] In step 315, the client-side module 174 determines that the current element is not valid if there are no more parent elements. In this case, processing proceeds from step 315 to step 365. In step 365, the client-side module 174 examines the Web page selection indicator to determine if text is selected. In one embodiment, the client-side module 174 determines if text is selected by getting the IHTMLSelectionObject pointer from the document. If it exists and has the type "text", the text has been selected.

[0052] Returning to the description of method 300, processing than proceeds from step 365 to step 370. In step 370, if the client-side module 174 determines that text is selected, processing proceeds to step 375. In step 375, the selected text is saved and the state that the text was selected is also saved. In an embodiment, the client-side module 174 saves the selected text by creating an IHTMLTxtRange for the selection. Then, the client-side module 174 calls get_text() on the range to obtain the actual text that is selected. Next, the text and the state that there is TEXT selected are saved for later use. Returning to decision step 370, if there is no text selected the method 300 returns to method 200 at step 245.

[0053] To further illustrate the method 300 of determining what element is selected by a user, five examples are described below in relation to an exemplary HTML element structure 400, illustrated in FIG. 4. The HTML code (with reference numbers added) that is represented by the structure 400 is as follows:

```
1- <html>
2- <body>
3- <a href="http://www.yahoo.com/">Yahoo!</a>
4- <br><br>
5- <a href="http://www.yahoo.com/"><img src=img.gif></a>
6- <br><br>
7- <table cellpadding=5 width=100%>
8- <tr bgcolor=red>
9- <td width=50% align=center bgcolor=#00ff00>
10- <a href="http://www.yahoo.com/">Yahoo!</a>
11- </td>
12- <td width=50% align=center bgcolor=#00ffff>
13- <span>
14- <font face=verdana>
15- <a href="http://www.yahoo.com/">Yahoo!</a>
16- </font>
17- </span>
18- </td>
19- </tr>
```

```
20- <tr height=25 bgcolor=red>
21- <td>hello</td>
22- </tr>
23- </table>
24- </body>
25- </html>
```

[0054] The first example illustrates a case when a user clicks on a text link. Line 3 of the exemplary HTML code shown above is a text link, and is represented in FIG. 4 by node 405. The element that was selected (clicked on) by the user, i.e., the source element, is an anchor, so it is marked as an anchor and the client-side module 174 walks up the HTML structure to the next parent element. The next parent element is the "body" element, node 410, which, in the present embodiment, is not an element type of interest. Therefore, it is ignored and the next parent element is identified. The next parent element is NULL, so the client-side module 174 proceeds to check if any text was selected. The type clicked on by the user is determined to be ANCHOR.

[0055] The second example illustrates the case when a user clicks on an image that is a link. Line 5 of the exemplary HTML code shown above is an image that is a link. It is represented in FIG. 4 by nodes 415 and 420. The element that was clicked on by the user, i.e., the source element, is an image, node 415, so it is marked as an image and the client-side module 174 walks up the HTML structure to the next parent element. The image is not an image map so no extra steps, with respect to the image, are taken. The next parent element is an anchor, node 420, so it is marked as anchor and the next parent element is identified. The next parent element is the "body" element, node 410, which is not an element type of interest. Therefore it is ignored and the next parent element is identified. The next parent element is NULL, so the client-side module 174 proceeds to check if any text was selected. The type clicked on by the user is determined to be ANCHOR and IMAGE.

[0056] The third example illustrates the case when a user clicks on a link with a green background. Lines 9 and 10 of the exemplary HTML code shown above is a link with a green background. It is represented in FIG. 4 by nodes 425 and 430. The element that was clicked on by the user, i.e., the source element, is an anchor, node 425, so it is marked as an anchor and the client-side module 174 walks up the HTML structure to the next parent element. The next four parent elements are "td", node 430, "tr", node 435, "table", node 440, and "body", node 410. None of these are element types of interest. Therefore they are ignored and the next parent element is identified. The next parent element is NULL, so the client-side module 174 proceeds to check if any text was selected. The type clicked on by the user is determined to be ANCHOR.

[0057] The fourth example illustrates the case when a user clicks on a link with a cyan background. Lines 12 and 18 of the exemplary HTML code shown above is a link with a cyan background. It is represented in FIG. 4 by nodes 450 to 465. The element that was clicked on by the user, i.e., the source element, is an anchor, node 450, so it is marked as an anchor and the client-side module 174 walks up the HTML structure to the next parent element. The next six parent elements are "font", node 455, span, node 460, "td", node 465, "tr", node 435, "table", node 440, and "body", node 410. None of these are element types of interest. Therefore they are ignored and the next parent element is identified. The next parent element is NULL, so the client-side module 174 proceeds to check if any text was selected. The type clicked on by the user is determined to be ANCHOR.

[0058] The fifth example illustrates the case when a user clicks on text with a red background. Lines 20 and 22 of the exemplary HTML code shown above is text with a red background. It is represented in FIG. 4 by nodes 470 to 475. The element that was clicked on by the user, i.e., the source element, is "td", node 470, this is not an element of interest so it is ignored. The next three parent elements are "tr", node 475, "table", node 440, and "body", node 410. None of these are element types of interest. Therefore they are ignored and the next parent element is identified. The next parent element is NULL, so the client-side module 174 proceeds to determine if any text was selected. The type clicked on by the user is determined to be NONE.

[0059] In alternate embodiments, various other schemes for determining which types of elements are of interest and which are to be ignored may be implemented.

[0060] FIG. 5 illustrates an exemplary embodiment of the method 500 for providing computer code for controlling a context menu. To use the additional context menu items of the present invention, a user obtains a client-side module 174. In an exemplary embodiment, a user obtains a client-side module 174 by connecting with a predetermined Internet site as in step 505. Then, in step 510, the server monitors for requests from the client. When a request for a client-side module 174 is received, processing proceeds to step 515. In step 515, using the controlling program 122, the Internet site 126 creates or downloads a client-side module 174, to the user's computer 150. The client-side module may be a library file.

[0061] Once the client-side module is obtained by the user, processing returns to step 505 where the client-side module 174 connects with a predetermined Internet site. The predetermined Internet site can be Internet site 126, from which client-side module 174 is downloaded, or alternatively, the predetermined Internet site can be another site, e.g., Internet site 136. Processing proceeds to step 510 where the Internet site 126 monitors for requests from the client. When a request for menu descriptions is received, processing proceeds to step 520 where the Internet site transmits the description of the menu items to the client. Then the Internet site 126 returns to step 510, where it monitors for requests from the client.

[0062] An embodiment of the invention may be provided as a feature of a method of controlling an Internet browser interface displayable by an Internet browser on a display of a computer, and enabling a user of the computer and Internet browser to access and navigate the Internet and to receive and display on the computer display one or more Web pages from one or more Internet sites, including the display of a Web page from a predetermined Internet site, the Internet browser having at least one Internet browser toolbar having at least one toolbar button providing a predetermined functionality to the user of the computer and Internet browser, the method can comprise, providing, at the predetermined Internet site, access to a program for controlling the Internet browser interface and making available for downloading by the predetermined Internet site, a file for causing the display of a persistent user toolbar adjacent to said Internet browser toolbar so as to create the visual impression that the user toolbar is an integrated part of the Internet browser, the user toolbar making additional functionality that is not part of the Internet browser prior to download available to the user after download as part of the

Internet browser interface, such that once the user toolbar is displayed the user toolbar remains displayed and said additional functionality remains available to the user regardless of a subsequent Internet site to which the Internet browser is caused to navigate after download. An Internet user may customize the browser so that each time the user accesses the Internet using the browser, user-defined information and/or functionality will be displayed with the browser interface, e.g., as a toolbar. The toolbar may include bookmarks, address and phone books, personal financial information, personalized news, and various functionality such as is available via ActiveX control and Plug-ins. In addition, if an Internet user has an account with a content provider, that user's specific account information (e.g., investment portfolio, news headlines, bookmarks, address book, additional context menu items etc.) may be dynamically displayed by the browser.

[0063] In an embodiment, user specific account information may include personalized context menu items. In such an embodiment, a user can choose which context menu items he or she desires to have, e.g., from a list of provided items. Those preferences can be stored with the content provider and/or locally at the client computer.

[0064] In addition, users with accounts can set up and use their additional context menu items in a first location and then have the same context menu settings at a second location without having to re-enter their personal settings. This can be done by storing their context menu settings at the content provider. The personal user settings can then be retrieved from any location the user wants to his or her personalized context menus.

[0065] In one exemplary example, wherein the method of controlling a context menu is implemented with a client-side module 174 for a browser toolbar, the description of the menu items are sent down, as a button, in the same feed from the Internet site 126 as the regular buttons for toolbar. This button "qs", which stands for QuickSearch, is marked as invisible so that the normal toolbar bar does not render it. Each menu item is a sub-item of the "qs" button and each item has certain pieces of information, e.g., type, URL, title, and optionally icons, which are downloaded in the background like other toolbar icons, mnemonic keys, a default URL to use if the user is not signed in and starting positions for different click types.

[0066] Also optionally sent down in the feed is a value indicating that the context menu feature should be disabled. Using this value, it is possible to turn this feature off, e.g., if there are complications or problems while implementing control of the context menu.

[0067] In addition, by having the additional context menu items described in the toolbar feed, different choices for different countries or users can be implemented. Furthermore, it is possible to dynamically add new items without requiring the user to download a new client-side module 174. The description of the "qs" button is stored on the client computer 150 using the caching mechanisms of the client-side module 174, so the user only has to download a new context menu item description if the items have changed.

[0068] An embodiment can also be implemented with information stored on the user (or client) computer. For example, context menu items can be stored on the client computer. When a user calls for a context menu, these stored context menu items are added to the displayed context menu. In addition, information stored on a client computer by other applications may be used to add additional features to the invention.

For example, information such as partner code, cobrands and the location at which a user downloads the invention, may be stored on a client's computer. This stored information can be used to determine the contents of the context menu. In an embodiment, a registry would be examined to determine which context menu items to use. In another example, one menu item can be, "email this link to . . .." Floating a mouse pointer over and/or selecting this item, creates a submenu including addresses from another application, e.g., Outlook. Additional features can also be implemented in accordance with the invention by adding additional computer code, e.g., plugins.

[0069] FIGs. 6-9 illustrate exemplary context menus for different clicked elements, implemented according to one embodiment of the invention. Referring to FIG. 6, browser 600 is a browser implemented according to one embodiment of the present invention. Browser 600 includes a plurality of windows, each providing various functionality to the Internet user. The browser 600 may comprise a first application window 605 that typically defines the general size, color, and layout of the browser 600 and includes window control buttons 670 (e.g., minimize, close, etc.) for that application window 605. The browser 600 may also comprise a browser window 615 and toolbar windows 610. The browser window 615 and the toolbar windows 610 typically define information and/or functionality that will assist an Internet user when accessing and navigating the Internet. For example, the browser window 615 and the toolbar windows 610 may provide toolbars, pull-down menus, Plug-ins, applications, etc.

[0070] For example, the toolbar windows 610 provided at the top (see FIG. 6) of the application window 605 define four toolbars 625, 630, 635, 640, which may include a variety of interface controls such as, for example, pull-down menus, functional buttons (e.g., stop, back, forward, home, etc.), and a combination of functional buttons and windows (e.g., a search button and window). The upper most toolbar 625 provides a plurality of pull-down menus; the second toolbar from the top 630 provides a plurality of functional buttons; the third toolbar from the top 635 provides a pull-down menu and a window, e.g., a URL address window; the bottom most toolbar 640 is implemented by the client-side module 174 and provides added functionality to the browser such as a search field and the context menu of the present invention. In an embodiment of the invention the context menu items descriptions may be included with the bottom most toolbar 640 as an invisible button. A browser window 615 is also provided as part of the browser 600 within which content from an Internet content provider 120 (see FIG. 1) may be displayed.

[0071] A user can call a context menu by, for example, right clicking on a mouse button. In FIG. 6, a user has right clicked while the pointer was on the background of the Web page, calling context menu 645. In this exemplary embodiment, the invention adds context menu items 650 based on the user clicking the background. The items were placed as the second set of menu items, and include icons and mnemonic keys. The context menu items include, "Add This Page To Yahoo! Bookmarks,". "IM this Page To Friend", and "Email This Page to Friend". As mentioned above, these menu items may be modified through the context menu items description.

[0072] In FIG. 7, a user has right clicked while the pointer was on the text link "Yahoo!" 705, calling context menu 710. In this exemplary embodiment, the embodiment adds context menu items 715 based on the user clicking "Yahoo!" 705. The items were placed as the second set of menu items, and include icons and mnemonic keys. The context menu items include, "Add This Link To Yahoo! Bookmarks," "IM this Link To Friend," and "Email This Link to Friend." These context menu items are different from the previous example.

[0073] In FIG. 8, a user has right clicked while the pointer was on the image link "Yahoo!Companion" 805, calling context menu 810. In this exemplary embodiment, the present invention adds context menu items 815 based on the user clicking "Yahoo!Companion" 805. The items were placed as the second set of menu items, and include icons and mnemonic keys. The context menu items include, "Add This Link To Yahoo! Bookmarks," "IM this Link To Friend," and "Email This Link to Friend."

[0074] In FIG. 9, a user has right clicked on the background while the text "Yahoo!" 905 was highlighted, calling context menu 910. In this exemplary embodiment, the present invention adds context menu items 915 based on the highlighted text and based on the user right clicking on the background. The items were placed as the second set of menu items, and include icons and mnemonic keys. The context menu items include, "Search the Web for Yahoo!," "Add This Page To Yahoo! Bookmarks," "IM this Page To Friend," and "Email This Page to Friend." If the "Yahoo!" 905 was highlighted and the user right clicked over a link, the context menu items would include "Add This Link To Yahoo! Bookmarks," "IM this Link To Friend," and "Email This Link to Friend" and "Search the Web for Yahoo!" (Not Shown).

[0075] Alternatively, the methods and system of controlling a context menu of the present invention can be implemented outside of the browser environment. The context menus used with the operating system of a computer and other applications, e.g., Microsoft Office, can be controlled in a similar way. For example, additional menu items can be added to the Windows start menu and the context menu called when a user right clicks or otherwise make a selection on the desktop, or the taskbar. Furthermore, additional menu items may be added to the right click menu used for Microsoft Word.

[0076] Thus, while there have been shown and described and pointed out fundamental novel features of the invention as applied to preferred embodiments thereof, it will be understood that various omissions and substitutions and changes in the form and details of the disclosed invention may be made by those skilled in the art without departing from the spirit of the invention. It is the intention, therefore, to be limited only as indicated by the scope of the claims appended hereto.